An FPGA-based custom high performance interconnection network

Mondrian Nüssle, Benjamin Geib, Holger Fröning, Ulrich Brüning
Computer Architecture Group
University of Heidelberg
Mannheim, Germany
e-mail: {mondrian.nuessle,benjamin.geib,holger.froening,ulrich.bruening}@ziti.uni-heidelberg.de

***Abstract*— An FPGA-based prototype of a custom high-performance network hardware has been implemented, integrating both a switch and a network interface in one FPGA. The network interfaces to the host processor over HyperTransport. About 85% of the slices of a Virtex IV FX100 FPGA are occupied and 10 individual clock domains are used. Six of the MGT-blocks of the device implement high-speed links to other nodes. Together with the integrated switch it is thus possible to build topologies with a node degree of up to 6, i.e. a 3D-torus or a 6D Hypercube. The target clock rate is 156 MHz with the links running at 6.24 Gbit/s and 200 MHz for the HyperTransport Core. This goal was reached with a 32-bit wide data path in the network-switch and link blocks. The integrated switch reaches an aggregate bandwidth of more than 45 Gbit/s. The resulting interconnection network features a very low latency – between nodes and including switching - close to 1 µs.**

I. INTRODUCTION

Networks for high performance computing (HPC) are typically optimized for high bandwidth. Less attention has been paid to latency, with some exceptions from research [1] and commercial vendors [2][3] featuring application to application latencies as low as 1.3 µs under certain conditions [4]. Specifically, the networks are not optimized for consistent small grain communications using small packets.

Within the EXTOLL research project, an interconnection network with optimized latency characteristics is developed. The design focuses on extremely low message latency and high message throughput. Both characteristics serve to increase the scalability of distributed applications running on a system that is connected using EXTOLL. These features are also important to enable the development and effective usage of message passing applications based on the *Message Passing Interface* (MPI) [5]. Also, EXTOLL is designed to support the emerging *Partitioned Global Address Space* (PGAS) [6] software paradigm, for example Universal Parallel C [7], very efficiently.

To verify the correctness of the design in addition to simulation and assess the performance of the architecture, prototyping in reconfigurable logic has been applied. With this prototype it becomes possible to test and analyze the interconnection network in a complete system environment running parallel programs, as for example common MPI benchmarks and open-source codes. These tests on an 8 node cluster have already given additional insights to the results gained from simulation.

The result is an HPC interconnection network with dedicated support to efficiently transport small packets, which is implemented completely in a single FPGA chip. A number of building blocks of the architecture have already been presented. Namely these are the host interface and the board used to connect the network to the host processor [8] , as well as the VELO engine [9] for small packet transfer and the RMA engine for zero-copy transfer of bulk data [10].

Figure 1 shows the network interface card which is based on the HTX-Board carrying a single Virtex IV FPGA [11]. It integrates both network interface and interconnection switch. The integrated switch renders additional centralized switches unnecessary. Six links support direct network topologies like 3D torus, 6D Hypercube or any other topology based on a node degree of up to 6. A HyperTransport [12] interface allows for direct connection to the host CPU, typically an AMD Opteron processor [13].

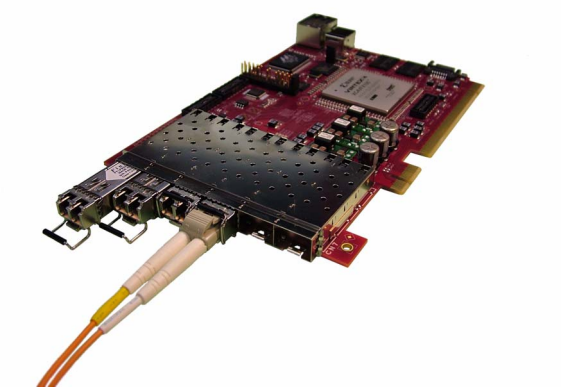

Figure 1. HTX-Board Revision 1.3

The main contribution of this work is to show the development of a complete interconnection network based solely on FPGA technology. Even using FPGA technology the performance results are competitive to designs based on ASIC technologies [9][10]. It also shows that modern FPGAs allow integrating a high-performance network

interface together with a switch on a single reconfigurable chip, although both an efficient architecture and a highly optimized implementation are necessary.

Due to the reconfigurable technology the complete network can be modified to special requirements or to research new architectural features.

The rest of this paper is structured as follows. In the next section the EXTOLL design with its different major blocks, the clock domains and major hardware requirements is shortly introduced. In section 3 the special provisions for the FPGA prototype and the resulting FPGA design are described. Section 4 presents the results and Section 5 related work. Finally, a conclusion is given in Section 6.

II. THE EXTOLL DESIGN

The EXTOLL design can be divided into three different logical blocks, the host-interface, the network interface controller (NIC) hardware and the network itself. The top-level block diagram of EXTOLL is shown in Figure 2. The host-interface is drawn in green, the NIC in blue and the network in red.

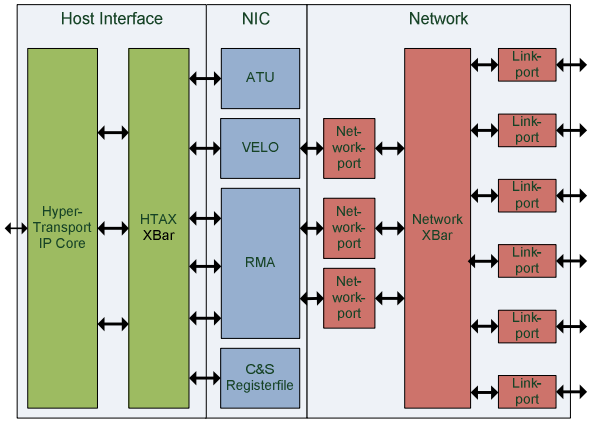

Figure 2. EXTOLL Top-Level Blockdiagram

Within the host-interface, there are two major blocks, the HT-Core and the on chip crossbar named HTAX. The HT-Core is described in detail in [8]. It actually uses three clock domains: two regional clocks for the source synchronous part of each 8 bit of the incoming 16-bit HyperTransport link. The transmit part as well as most of the RX path uses a common core clock which is an integral divider of the link clock. In this case a 200 MHz core clock together with a 400 MHz link clock implement an HT400 interface that is capable of a unidirectional bandwidth of 1.6 GByte/s. Between the HT-Core and the HTAX, a number of synchronizing FIFOs are used for the transition from the HT-Core clock to the EXTOLL core clock, which is here chosen to be 156 MHz.

The HTAX crossbar is a parametrizable IP block developed at the Computer Architecture Group as an on-chip network. Here it connects the HT-Core with the different Functional Units (FUs) that form the NIC block. The NIC block resides completely within the EXTOLL core clock domain, as well as the network ports and the EXTOLL crossbar. Within the NIC, the Address Translation Unit (ATU) module implements a virtual-to-physical address translation for zero-copy network transactions (not unlike a MMU in a processor), VELO is a unit which implements extremely low send/receive communication semantics [9] and RMA [10] implements a low-overhead remote direct memory access (RDMA) which can be used for larger messages or bulk transfers. The Control & Status Register of EXTOLL provides all the necessary registers to interface with the software running on the host. The Networkport converts an internal protocol to the network protocol. It converts from the NIC data-width of 64 bit to the network data path width, which is 16 or 32 bit for the prototype implementation. The EXTOLL crossbar forms the switch of the EXTOLL architecture. It consists of fully parametrizable RTL code plus a number of scripts to facilitate implementation of crossbars with different number of ports and data-path widths. In the prototype a 9 port switch is implemented with 3 ports connected to the internal FUs and 6 ports to the external links. The crossbar implements virtual output queuing, credit based flow-control and cut-through switching. As a result it features lossless packet transport, a low latency and some resilience against head-of-line blocking.

The link ports implement a hardware retransmission protocol, framing, 8b10b coding of the data stream and the lower link layers using the MGT block of the Virtex IV device. Each link port has one regional clock domain for the incoming path, before the data is synchronized into the core clock domain using a synchronizing FIFO.

The design uses numerous RAM blocks, notably some tables in the registerfile, 3 queues in the RMA unit, one RAM block in each crossbar input port for the virtual output queues and for all the synchronizing FIFOs in the design.

III. FPGA CONSIDERATIONS

For EXTOLL the Virtex IV FPGA [14] on the HTX-Board represents a very challenging target architecture. The HT-Core is required to run at 200 MHz internal frequency and the EXTOLL core frequency is required to be at least 156 MHz, which is the lowest frequency where the same clock source can be used for the core frequency and the MGT reference clock [15].

Since the HTX board is only a prototyping platform for EXTOLL, all FPGA specific modules and primitives have been encapsulated in wrapper modules, in order to be able to easily port the design from one FPGA platform to another FPGA or ASIC platform or vendor. Using these wrapper modules eases the transition when changing platforms, since only these modules need to be adapted. Therefore the design is not hand tailored to the Virtex IV, but kept as generic as possible. Although all optimizations offered by the tool chain have been applied, several area and timing optimizations are necessary, since the original design required more than 100 percent of the available logic and did not meet the timing constraints. The following sections describe what has been done to maintain portability, minimize area and reach timing closure.

A. Input/Output Considerations

The Xilinx MGTs are not designed for low latency communication. Instead of using the integrated word alignment and 8b/10b coding, external modules are used in the FPGA fabric, in order to reduce the latency inflicted by the transceivers. Bypassing all these internal structures reduces the latency of the MGTs by more than 44% [16] .

Since the HT400 interface uses double data rate, 800 Mb/s I/Os are needed for the communication with the Opterons. Therefore the Xilinx ISERDES and OSERDES blocks are used in combination with the integrated IDELAY function to find the eye for data sampling and the BITSLIP function for lane de-skewing.

B. RAM blocks

For the RAMs, a wrapper module has been written from which the Xilinx XST synthesis tool [17] can infer BlockRAMs without explicitly instantiating Xilinx specific primitives. This has been done solely for portability reasons.

C. FIFOs

The design uses many FIFOs as storage and management solution, so finding an efficient FIFO implementation is necessary.

For small FIFOs up to a depth of 16 stages an area efficient implementation based on Xilinx SRLs [18] is chosen. For larger FIFOs SRLs are no longer efficient and BlockRAMs are used instead.

D. Round-Robin-Arbiter Optimizations

The round robin arbiters used in the design are based on the implementation described in [19]. Internally it uses a thermometer code based encoding. For the grant generation this thermometer code has to be converted into either a one-hot encoding or a binary encoding. In the initial design the binary encoding has been generated using the thermometer code in a Verilog *case* statement (column "case-based, binary grant" in Table I). By replacing the *case* statement with a *casex* and writing the statements as a *full and parallel case* the area and timing significantly improved (see column "casex-based, binary grant" in Table I).

TABLE I. OPTIMIZATIONS ON THE 36 INPUT RR-ARBITER

Resource	*case*-based, binary grant	*casex*-based, binary grant	*casex*-based, one-hot grant
Slices	248	178	169
FlipFlops	79	81	72
LUTs	439	308	272
fmax	120.7 MHz	188.7 MHz	> 250 MHz

Avoiding the binary coded grant and instead using a one-hot coded grant improved timing and area even more, since the thermometer to one-hot conversion can be implemented more efficient than thermometer to binary conversion. Column "casex-based, one-hot grant" of Table I shows the improvement of the arbiter when avoiding binary coded grants. Additional performance improvements apply for modules depending on the grant signals.

E. Multi-queue FIFO Implementation and Optimization

As already mentioned before, the EXTOLL crossbar is a virtual output queued switch. It uses multiple FIFOs to manage the packets for the different destinations and virtual channels. EXTOLL uses a crossbar with 9 output ports with 4 virtual channels each, which results in a total of 36 queues and therefore 36 pairs of pointers.

Using a separate FIFO for each queue is not applicable since this would either require more BlockRAMs than available on the FPGA or many slices (column "Multiple FIFOs" in Table II). Instead, these FIFOs are managed in a multi-queue structure, which consists of one BlockRAM with multiple read and write pointers.

This results in many registers and multiplexing structures. The required resources are summarized in the column "Register based pointer" of Table II. Using distributed RAM to hold the pointer structures should reduce the size significantly. The shift-in side of the multi-queue FIFO may process a different queue than the shift-out side in any given cycle; hence two read-ports are necessary for both read and write pointer. One write port is sufficient since write and read pointers are only updated by the shift-in respectively the shift-out side. Unfortunately, a three port RAM with one write and two read ports is not available.

TABLE II. OPTIMIZATIONS ON THE MULTIQUEUE-FIFO

Resource	Multiple FIFOs	Register based pointer	RAM based pointer
Slices	1946	1796	614
FlipFlops	871	1002	209
LUTs	1946	1738	1133
BlockRAMs	0	2	2
Distributed RAM	504	0	224
fmax	197 MHz	116 MHz	267 MHz

The solution chosen is to use four distributed RAMs, two in parallel for the read pointers and two in parallel for the write pointers. By using two RAMs in parallel, one write

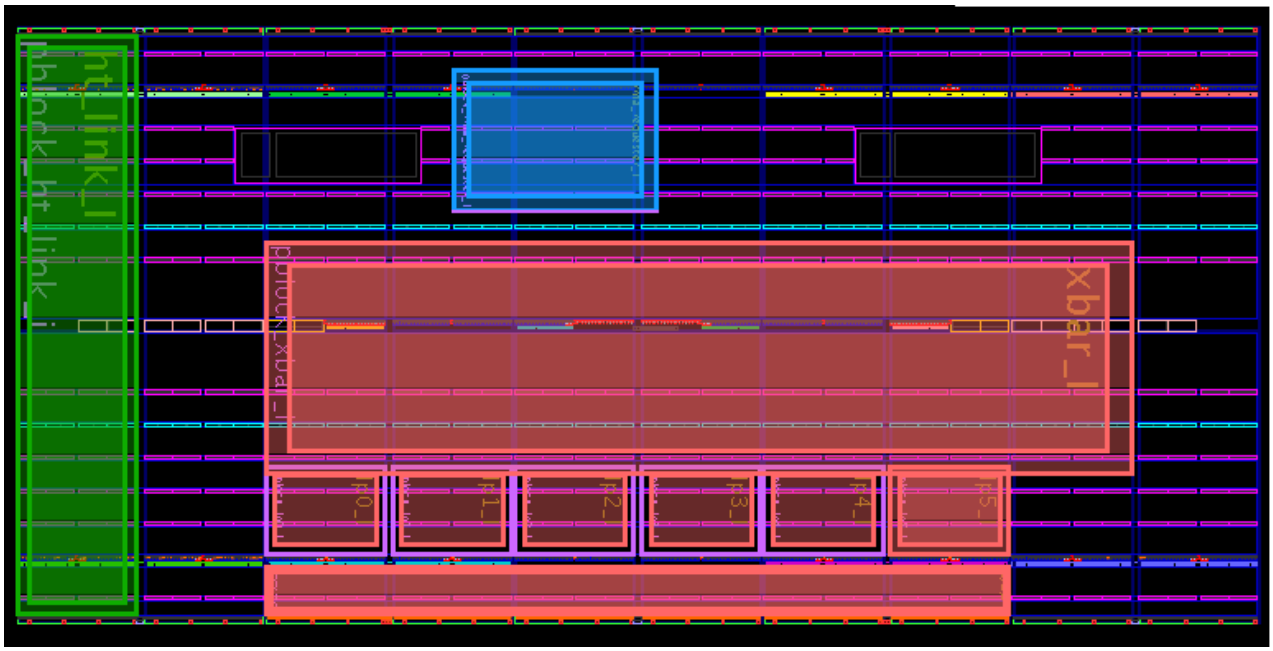

Figure 3. EXTOLL Floorplan

port and two read ports can be realized. The write bus is connected to both RAMs, and the second port of each RAM realizes one of the read ports of the structure. The last column of Table II ("RAM based pointer") shows the amount of resources used when holding the pointer structures in distributed RAM instead of registers.

In order to be able to update the RAM based pointer structure it is necessary to take the additional read latency of the pointer RAM into account to ensure correctness. But in EXTOLL the maximum shift-in rate for this FIFO is one word every four clock cycles, thus this problem is avoided.

F. Floorplanning

In the first iterations only a bounding box around the HT-Core was necessary to reach the needed 200 MHz in its clock domains (the green bounding box in Figure 3). By adding more and more features to the EXTOLL design and therefore pushing the utilization of the chip to a maximum, a more detailed floorplan became necessary to be able to reach the goal of 156 MHz in the EXTOLL clock domains. The big red box is the crossbar switch accompanied by the smaller red boxes which hold the 6 link ports.

The red box at the bottom of the floorplan is the wrapper module for the Xilinx MGTs, the serial transmitters. The blue box is a module belonging to the RMA functional unit, which turned out to be very timing critical. Noticeable here is that the box floats in the middle of the device. This is due to the fact that there are numerous unconstrained (in respect to bounding boxes) modules around the RMA that need to fit between itself and the red XBAR bounding box.

Giving a module too much 'space' on the FPGA proved to be counterproductive when trying to reach high frequencies. In the end timing closure has been reached by adjusting the size of the bounding boxes and their locations in an iterative process using Xilinx PlanAhead.

IV. RESULTS

On both speed-grade 12 and speed-grade 11 devices, the target frequency of 156 MHz has been achieved for the first variant with an internal data width of 16 bit on the network layer using the afore mentioned techniques to reduce area and improve timing. Table III shows the utilization of the device for this design after all optimizations.

TABLE III. UTILIZATION OF FX100 USING 16BIT DATA WIDTH

Resource	Usage total	Usage in %
Slices	35,162	83%
FlipFlops	27,233	32%
LUTs	61,166	72%
BRAMs	119	31%

The final placement of the design is shown in Figure 4. The color coding is the same as in Figure 3. The green primitives outside the HT-Core bounding box belong to the HTAX on-chip network which connects the HT-Core with the functional unit. These units are all marked blue. The red dots outside the bounding boxes belong to the network ports. The yellow dots in the final placement belong to an I^2C module, which configures a programmable oscillator [20], reads data from an external EEPROM and controls the optical SFP transceivers.

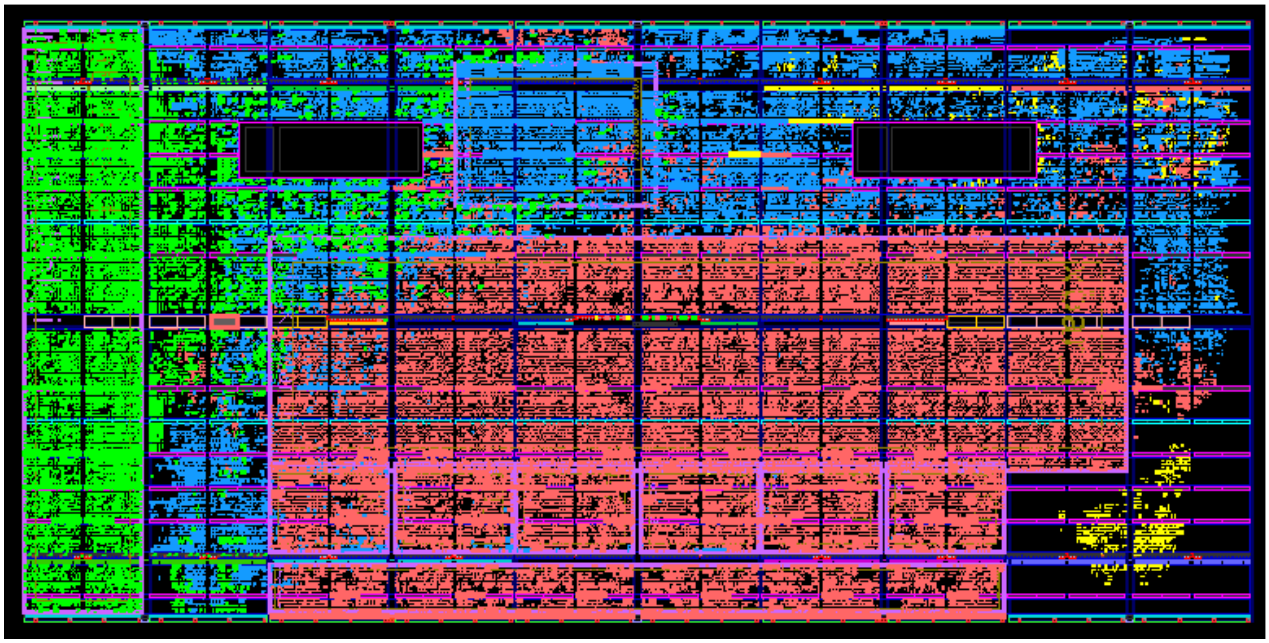

Figure 4. EXTOLL Post Map Placement

This design has a half-round-trip latency of 1.18 µs using the VELO functional unit with 8 bytes payload. Maximum throughput is 267 MByte/s using RDMA put operations of the RMA functional unit with 4 kByte payload.

In a second variant, EXTOLL is implemented with a data width of 32 bit for the network part (the NIC part is always 64 bit wide) on a speed-grade 12 device. This results in a serial transmission rate of 6.24 Gbit/s. Table IV shows the utilization for this design.

TABLE IV. UTILIZATION OF FX100 USING 32BIT DATA WIDTH

Resource	Usage total	Usage in %
Slices	35,985	85%
FlipFlops	28,442	33%
LUTs	63,382	75%
BRAMs	140	37%

The half round trip latency for the 32 bit design is 1.16 µs. The maximum throughput is 552 MByte/s. The latency in the 32 bit case is lower due to the fact that the serialization between functional units and network layer uses only a factor of 2 instead a factor of 4 as it is the case using 16 bit data width.

The design is dominated by control logic, only a small portion of the used resources is needed for data paths. This can be seen by comparing Table 4 and Table 5.

V. RELATED WORK

There have been a number of FPGA-based network architectures in the past. Examples are a SCI implementation using FPGAs [21] and the Clint network [22]. In more recent times, a prototype for DIMMnet-2 network interface controller has been built using a Virtex II Pro 70 FPGA device [23]. The prototype implements only limited functionality of the complete design, for example only packets of up to 496 bytes can be processed. The core logic runs with 98 MHz in this design and about 42% of the slices are used. For the network, the controller interfaces to an Infiniband fabric. In an evaluation, the NIC showed an excellent latency of about 1 µs, but with an on-chip direct loopback, i.e. without the latency from the actual link and physical layer and without the latency of a switch. In contrast, the 1.16 µs of latency for EXTOLL does include all these latencies, an internal loopback using the EXTOLL switch (thus still including one level of switching) accounts for approximately 800 ns.

In [24] a NIC is described that is able to perform RDMA write operations only. It is implemented also on a Virtex II device and runs with 100 MHz for the core and 78 MHz for the links.

For the JNIC project [25] a Gigabit-Ethernet NIC is implemented using an Altera Stratix EP1S80 FPGA attached to the Front-side bus of an Intel Xeon machine.

FPGAs have also been used to build entire parallel machines, an example can be found in [26] for which also a MPI implementation exists [27].

VI. CONCLUSION AND FUTURE WORK

There was a number of optimization necessary to reach timing closure for the EXTOLL design on a Virtex IV FPGA and fit the design into the available device. Using careful optimization of pipeline stages plus RTL coding techniques that fit high-speed synthesis are the most important factors. Also, the usage of all of the features and hard macro blocks of the underlying FPGA technology proved to be helpful. For example to use SRL16 blocks for true first-word-fall-

through FIFOs saves many slices. In the last stage, Xilinx PlanAhead tool was used to perform floor planning and actually get the last performance out of the design that was necessary to reach timing closure.

These optimizations allow implementing a high performance interconnection network with FPGAs. The results are very encouraging, as already reported in [9] and [10]. These performance numbers are competitive to widely available high performance network solutions, especially considering the fact that an FPGA implementation is compared against ASIC implementations. Additionally, the switch is already included.

The design is actively used to develop all of the software components to support a complete system suitable to run parallel applications; both using the message passing (MPI) and shared memory (PGAS) paradigm.

Currently an 8 node development cluster with 32 CPU cores is running in our lab and serves as a platform for more complex tests and benchmarks.

A powerful cluster with 1024 cores at the Universidad Politècnica de Valencia is being equipped with HTX-Boards and will allow intense testing of the system's scalability. Insights from these tests will be used to improve the overall architecture and reconfigurability will be exploited to explore new approaches.

In the future, a migration to a new FPGA device - probably of the 40 nm generation - is planned. This device will offer additional logic capacity which can be used to bring the network data path width to 64 bit. This widens the external links using additional transceivers and therefore increases the bandwidth. Furthermore and maybe most important, it allows to add additional modules for example to support multicast and barrier operations or to increase fault tolerance.

ACKNOWLEDGEMENTS

We would like to express our thanks to all the people that contributed to the EXTOLL project and the FPGA prototype, especially Heiner Litz, Dr. David Slogsnat, Sven Schenk, Niels Burkhardt, Alexander Giese and Martin Scherer. We also thank Xilinx Inc. for their generous support.

REFERENCES

[1] Fröning H., Nüssle M., Slogsnat D., Haspel P.R., and Brüning U., Performance Evaluation of the ATOLL Interconnect, IASTED Conference, Parallel and Distributed Computing and Networks (PDCN), Innsbruck, Austria, Feb. 15 - 17, 2005.

[2] Myrinet Myricom Inc. Myrinet Express (MX): A High-Performance, Low-Level Message-Passing Interface for Myrinet, version. 1.2. 2006.

[3] Infiniband Trade Association. InfiniBand Architecture Specification Volume 1, Release 1.1, 2002.

[4] Sur S., Koop M. J., Chai L., Panda D. K., Performance Analysis and Evaluation of Mellanox ConnectX InfiniBand Architecture with Multi-Core Platforms, 15th Annual IEEE Symposium on High-Performance Interconnects (HOTI 2007), August 2007.

[5] Message Passing Interface Forum, MPI: A Message-Passing Interface Standard Version 1.3, 2008

[6] Yelick K., Bonachea D., et al., Productivity and performance using partitioned global address space languages, In Proceedings of the 2007 International Workshop on Parallel Symbolic Computation, London, Canada, July 2007.

[7] Bonachea D., GASNet specification, v1.1, Technical Report UCB/CSD-02-1207, U.C. Berkeley, October 2002.

[8] Slogsnat D., Giese A., Nüssle M., and Brüning U., An Open-Source HyperTransport Core, ACM Transactions on Reconfigurable Technology and Systems (TRETS), Vol. 1, Issue 3, p. 1-21, Sept. 2008.

[9] Litz H., Fröning H., Nüssle M., and Brüning U., VELO: A Novel Communication Engine for Ultra-low Latency Message Transfers, 37th International Conference on Parallel Processing (ICPP-08),Portland, Oregon, USA, 2008.

[10] Nüssle M., Scherer M., and Brüning U., A resource optimized remote-memory-access architecture for low-latency communication, accepted for publication in 38th International Conference on Parallel Processing (ICPP-09), Vienna, Austria, 2009.

[11] Fröning H., Nüssle M., Slogsnat D., Litz H., and Brüning U., The HTX-Board: A Rapid Prototyping Station, 3rd annual FPGAworld Conference, Nov. 16, 2006, Stockholm, Sweden.

[12] Hypertransport Consortium, HyperTransport Technology I/O Link - White Paper, www.hypertransport.org, July 2001.

[13] Advanced Micro Devices (AMD), AMD Opteron Product Data Sheet, Publication #23932, 2004.

[14] Xilinx Corporation, Virtex-4 User Guide, Document ug070 v1.5, www.xilinx.com, 2006.

[15] Xilinx Corporation, Virtex-4 RocketIO Multi-Gigabit Transceiver User Guide, Document ug076 (v3.2), www.xilinx.com, 2006

[16] Schenk S., Architecture Analysis of Multi-Gigabit-Transceivers for Low Latency Communication, Diploma Thesis at the Computer Architecture Group of the University of Mannheim Germany, 2008.

[17] Xilinx Inc., XST User Guide, 2008.

[18] Gopalakrishnan L., FIFOs Using Virtex-II Shift Registers, Xilinx Inc., Application Note XAPP256 (v1.3) January 5, 2005.

[19] Savin, C.E., McSmythurs, T., and Czilli, J., Binary tree search architecture for efficient implementation of round robin arbiters, IEEE International Conference on Acoustics, Speech, and Signal Processing, Montreal, Quebec, Canada, 2004.

[20] Silicon Laboratories, SI570/571 Any-Rate I2C Programmable XO/VCXO, Revision 0.31, http://www.silabs.com, 2007

[21] Trams M., and Rehm W., SCI Transaction Management in our FPGA-based PCI-SCI Bridge, Proceedings of SCI Europe, Dublin, 2001.

[22] Fugier N., Herbert M., Lemoine E., and Tourancheau B., MPI for the Clint Gb/s Interconnect, Recent Advances in Parallel Virtual Machine and Message Passing Interface, Vol. 2840/2003, Lecture Notes in Computer Science, Springer, Heidelberg, 2003.

[23] Tanabe N., Kitamura A. et al., Preliminary evaluations of a FPGA-based-prototype of DIMMnet-2 network interface, IEEE International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems, Oahu, Hawaii, USA, 2005.

[24] Marazakis M., Xinidis K., Papaefstathiou V., and Bilas A., Efficient remote block-level I/O over an RDMA-capable NIC, Proceedings of the ACM International Conference on Supercomputing (ICS), Cairns, Australia, 2006.

[25] Schlansker M., Chitlur N., et al., High-performance ethernet-based communications for future multi-core processors, In Proceedings of the 2007 ACM/IEEE Conference on Supercomputing (SC '07), Reno, Nevada, USA, 2007.

[26] Patel A., Madill C., Salda–a M., Comis C., Pom s R., and Chow P., A Scalable FPGA-based Multiprocessor, In IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'06), pages 111-120, 2006.

[27] Saldana M., and Chow P., TMD-MPI: An MPI implementation for multiple processors across multiple FPGAs, IEEE 16th International Conference on Field Programmable Logic and Applications, Madrid, 2006.